

**Tema**

INTRODUCCION A WPF

**Tutor**

Ing. Eduardo Mauricio Campaña Ortega

MIS.MDU.CCNA.CCIA.

PhD. (c) Ingeniería de Software

PhD. (c) Seguridad Información

Fecha

26/08/2023

Tabla de contenido

[1. INTRODUCCIÓN 3](#_Toc143980957)

[2. OBJETIVO 4](#_Toc143980958)

[2.1 OBJETIVO GENERAL 4](#_Toc143980959)

[2.2 OBJETIVOS ESPECÍFICOS 4](#_Toc143980960)

[3. MARCO TEÓRICO 4](#_Toc143980961)

[3.1 WPF 4](#_Toc143980962)

[3.2 XAML (LENGUAJE DE MARCADO EXTENSIBLE DE APLICACIONES). 4](#_Toc143980963)

[3.3 ENLACE DE DATOS EN APLICACIONES WPF. 5](#_Toc143980964)

[3.4 RECURSOS GRAFICOS 5](#_Toc143980971)

[4. DESARROLLO 6](#_Toc143980991)

[4.1 CREACIÓN DEL PROYECTO 6](#_Toc143980992)

[4.1.1 CREAR UNA APLICACIÓN LLAMADA TALLER\_INTRODUCCIONWPF\_GRUPO#. 6](#_Toc143980993)

[4.1.2 CREACIÓN DE LA INTERFAZ PRINCIPAL DEL PROYECTO. 8](#_Toc143980994)

[4.2 CREACIÓN DE VENTANA RECEPTORA 11](#_Toc143980995)

[4.2.1 CREACIÓN DE LA NUEVA VENTANA. 11](#_Toc143980996)

[5. EJECUCIÓN DEL PROYECTO 13](#_Toc143980997)

[6. CONCLUSIONES 14](#_Toc143980998)

[7. RECOMENDACIONES 15](#_Toc143980999)

[8. BIBLIOGRAFÍA 15](#_Toc143981000)

**ÍNDICE DE IMÁGENES**

[Figura 1. Logotipo identificativo de Visual Studio 2022 5](#_Toc124714550)

[Figura 3. Selección del tipo de aplicación a desarrollar 11](#_Toc124714552)

[Figura 4. Selección del nombre y directorio donde guardar el proyecto. 11](#_Toc124714553)

[Figura 5. Estructura del proyecto generado. 12](#_Toc124714554)

[Figura 6. Creación de carpetas MVC. 12](#_Toc124714555)

[Figura 7. Estructura del proyecto con MVC. 13](#_Toc124714556)

[Figura 8. Cambio de nombre del formulario para la Vista 13](#_Toc124714557)

[Figura 9. Estructura de la carpeta Vista 14](#_Toc124714558)

[Figura 10. Adición del menú Cuadro de herramientas 14](#_Toc124714559)

[Figura 11. Codificación archivo Vista 15](#_Toc124714560)

# INTRODUCCIÓN

La creación de aplicaciones de software robustas e interactivas es esencial en el panorama tecnológico actual. Windows Presentation Foundation (WPF) se ha destacado como un marco de desarrollo de interfaz gráfica de usuario (GUI) de Microsoft que permite la creación de aplicaciones de Windows con interfaces visuales atractivas y funcionales. En combinación con la potencia de Visual Studio, el entorno de desarrollo integrado de Microsoft, WPF ofrece a los desarrolladores las herramientas necesarias para construir aplicaciones modernas y atractivas.

**Windows Presentation Foundation (WPF):** es un marco de desarrollo de interfaz gráfica de usuario que forma parte del ecosistema de tecnologías de Microsoft .NET. WPF se centra en la separación entre la lógica de la aplicación y la presentación visual, utilizando el concepto de XAML (Lenguaje de Marcado Extensible de Aplicaciones) para definir la estructura de la interfaz gráfica. Esto permite a los diseñadores y desarrolladores colaborar de manera efectiva, ya que los diseñadores pueden trabajar en el diseño visual mientras que los desarrolladores implementan la lógica subyacente en lenguajes como C#.

**Visual Studio:** Es el entorno de desarrollo integrado (IDE) principal de Microsoft utilizado para crear una amplia gama de aplicaciones, desde aplicaciones de escritorio hasta aplicaciones web y móviles. Proporciona herramientas de desarrollo, depuración y diseño que simplifican el proceso de creación de software. En el caso de la creación de aplicaciones WPF, Visual Studio ofrece plantillas predefinidas que aceleran el proceso de inicio y proporcionan una plataforma para construir interfaces de usuario atractivas y altamente personalizables.

**Componentes Clave:** Al crear una aplicación WPF en Visual Studio, los elementos clave incluyen,

XAML (Lenguaje de Marcado Extensible de Aplicaciones): Este lenguaje basado en XML se utiliza para definir la estructura visual de la interfaz de usuario, incluyendo elementos como botones, ventanas, cuadros de texto y más.

Lógica de la Aplicación: Los desarrolladores implementan la lógica de la aplicación utilizando lenguajes de programación como C#. Esto incluye la manipulación de datos, interacciones con el usuario y la lógica empresarial subyacente.

Estilos y Plantillas: WPF ofrece la capacidad de definir estilos y plantillas reutilizables que permiten una coherencia visual en toda la aplicación y facilitan la personalización.

Enlace de Datos: WPF permite enlazar datos directamente a la interfaz de usuario, lo que significa que los cambios en los datos se reflejan automáticamente en la interfaz y viceversa.

Recursos Gráficos: Se pueden incorporar gráficos, imágenes y otros recursos multimedia en la aplicación para mejorar la experiencia visual y la interactividad.

En este informe, exploraremos en detalle cómo crear una aplicación WPF utilizando Visual Studio, abordando desde la configuración inicial hasta la implementación de la lógica y la personalización de la interfaz. A medida que profundicemos en los aspectos técnicos, comprenderemos cómo aprovechar la potencia de WPF y Visual Studio para crear aplicaciones modernas y atractivas que satisfagan las necesidades de los usuarios en el mundo digital de hoy.

# OBJETIVO

### OBJETIVO GENERAL

El objetivo general de este proyecto es desarrollar una aplicación WPF utilizando Visual Studio que proporcione una interfaz de usuario moderna e interactiva para satisfacer las necesidades de los usuarios, al tiempo que se aprovechan las capacidades de Windows Presentation Foundation para lograr una experiencia de usuario atractiva y funcional.

### OBJETIVOS ESPECÍFICOS

* Diseñar la Interfaz de Usuario.
* Integrar Recursos Multimedia.
* Aplicar Principios de Personalización y Usabilidad.

# MARCO TEÓRICO

## WPF

Windows Presentation Foundation (WPF) es un marco de desarrollo de interfaz gráfica de usuario (GUI) creado por Microsoft como parte de la plataforma .NET. WPF ofrece un enfoque moderno y versátil para crear aplicaciones de escritorio con interfaces visuales atractivas y altamente personalizables. A diferencia de las interfaces de usuario convencionales basadas en ventanas y controles, WPF se basa en el concepto de "escena gráfica" donde los elementos visuales se organizan en una jerarquía tridimensional.

## XAML (LENGUAJE DE MARCADO EXTENSIBLE DE APLICACIONES).

XAML, el Lenguaje de Marcado Extensible de Aplicaciones, es una piedra angular en el desarrollo de aplicaciones Windows Presentation Foundation (WPF). Es un lenguaje basado en XML que permite la definición de la estructura de la interfaz de usuario y otros recursos visuales en un formato legible por humanos y altamente estructurado. En este reporte, exploraremos en profundidad el papel y la funcionalidad de XAML en las aplicaciones WPF.

**Estructura y Componentes de XAML:** XAML se compone de elementos y atributos que representan objetos visuales y su configuración en la interfaz de usuario. Los elementos pueden anidarse para crear jerarquías visuales complejas. Cada elemento puede contener atributos que definen las propiedades del objeto, como su tamaño, posición, estilo y contenido. Esto permite a los desarrolladores y diseñadores colaborar de manera efectiva, ya que los diseñadores pueden trabajar en la estructura visual mientras que los desarrolladores definen la lógica subyacente en lenguajes como C#.

**Separación de la Presentación y la Lógica:** Uno de los beneficios clave de XAML es su capacidad para separar claramente la presentación de la lógica de la aplicación. Esta separación permite una división más eficiente de tareas entre los equipos de diseño y desarrollo. Los diseñadores pueden centrarse en la estética y la organización visual de la interfaz, mientras que los desarrolladores pueden concentrarse en la implementación de la funcionalidad y la interacción.

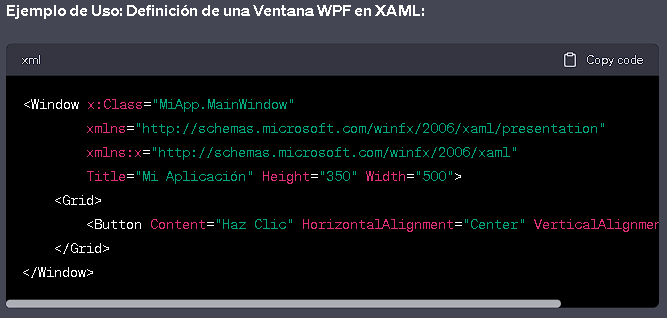


Figura 1. Ejemplo XAML

En este ejemplo, se define una ventana con un botón en el centro. El elemento `Window` define las propiedades de la ventana, como el título y las dimensiones. El elemento `Grid` es un contenedor que organiza los elementos en filas y columnas, y dentro de él, se encuentra el botón con su contenido y alineación.

## ENLACE DE DATOS EN APLICACIONES WPF.

## El enlace de datos es el proceso de conectar los datos, ya sean almacenados localmente o provengan de fuentes externas, a los elementos visuales en la interfaz de usuario. Esto permite que los cambios en los datos se reflejen automáticamente en la interfaz y viceversa, eliminando la necesidad de manipular la interfaz manualmente cada vez que los datos cambian. En las aplicaciones WPF, el enlace de datos se logra principalmente a través del lenguaje de marcado XAML.

## Tipos de Enlace de Datos:

## Enlace Unidireccional: Los datos se muestran en la interfaz, pero los cambios en la interfaz no afectan los datos subyacentes.

## Enlace Bidireccional: Los cambios en la interfaz se reflejan en los datos y viceversa. Es especialmente útil para formularios y entradas de usuario.

## Enlace a Elementos Ancestros: Permite acceder a datos en elementos superiores de la jerarquía visual, lo que es útil para controles dentro de contenedores.

## Enlace a Elementos Futuros: Permite establecer un enlace incluso antes de que los elementos de destino estén creados

## RECURSOS GRAFICOS

## Los recursos gráficos desempeñan un papel esencial en la creación de experiencias visuales ricas y atractivas en aplicaciones Windows Presentation Foundation (WPF). En este informe, exploraremos en profundidad cómo los recursos gráficos enriquecen las aplicaciones WPF y cómo se integran para mejorar la apariencia y la interactividad.

## Importancia de los Recursos Gráficos:

## Los recursos gráficos abarcan una variedad de elementos visuales, desde imágenes y iconos hasta gráficos vectoriales y animaciones. Estos elementos juegan un papel crucial en la presentación visual de la aplicación y contribuyen significativamente a la percepción y usabilidad de la interfaz de usuario.

## Tipos de Recursos Gráficos:

## WPF ofrece varios tipos de recursos gráficos que se pueden integrar en las aplicaciones:

## Imágenes: Las imágenes permiten la visualización de elementos estáticos. Pueden ser imágenes rasterizadas en formatos como JPEG o PNG, o gráficos vectoriales escalables (SVG) que mantienen su calidad independientemente del tamaño.

## Iconos: Los iconos son pequeñas representaciones visuales utilizadas para indicar acciones, estados o categorías en la interfaz de usuario. WPF admite la integración y el uso de iconos para mejorar la navegación y la comprensión.

## Gráficos Vectoriales: Los gráficos vectoriales permiten la creación de formas y elementos visualmente complejos que se escalan sin pérdida de calidad. WPF utiliza DirectX para representar gráficos vectoriales, lo que garantiza un rendimiento óptimo.

## Animaciones: Las animaciones añaden dinamismo y atractivo a la interfaz de usuario. WPF admite animaciones de propiedades, como el movimiento o el cambio de color, que se pueden aplicar a elementos visuales.

## Integración de Recursos Gráficos:

## La integración de recursos gráficos en aplicaciones WPF es flexible y efectiva:

## XAML: Los recursos gráficos se pueden definir directamente en XAML utilizando elementos como <Image> o <Path> para gráficos vectoriales. Esto permite una fácil inclusión y personalización de los recursos visuales.

## Rutas de Acceso: Las rutas de acceso permiten especificar la ubicación de recursos externos, como imágenes o archivos de iconos, para su uso en la aplicación.

## Estilos y Plantillas: Los recursos gráficos también se pueden incorporar en estilos y plantillas para garantizar una apariencia coherente en toda la aplicación.

## Beneficios de los Recursos Gráficos:

## La integración efectiva de recursos gráficos en aplicaciones WPF ofrece numerosos beneficios:

## Mejora de la Experiencia del Usuario: Los recursos gráficos mejoran la estética y la interactividad de la interfaz de usuario, lo que resulta en una experiencia más atractiva y agradable para el usuario.

## Comunicación Visual: Los iconos y gráficos vectoriales permiten una comunicación visual rápida y efectiva, facilitando la comprensión de las acciones y estados.

## Personalización: La utilización de recursos gráficos permite una personalización detallada de la interfaz de usuario, lo que satisface las preferencias y necesidades individuales de los usuarios.

# 4. DESARROLLO

# 4.1 CREACIÓN DEL PROYECTO

## 4.1.1 CREAR UNA APLICACIÓN LLAMADA TALLER\_INTRODUCCIONWPF\_GRUPO#.

Abra el IDE de desarrollo Visual Studio 2022.



Figura 2. Logotipo identificativo de Visual Studio 2022

Primero dentro del disco C y cree una carpeta llamada DISTRIBUIDAS allí crea otra subcarpeta llamada INTRODUCCION\_WPF, a su vez dentro de esta última crea 2 carpetas, una con el nombre de “Aplicativo” y otra con el nombre de “Documentación”. Una vez haya creado las carpetas abra Visual Studio 2022 y seleccionar crear nuevo proyecto, en el tipo de aplicación busca Aplicación de Windows Forms (.NET Framework).

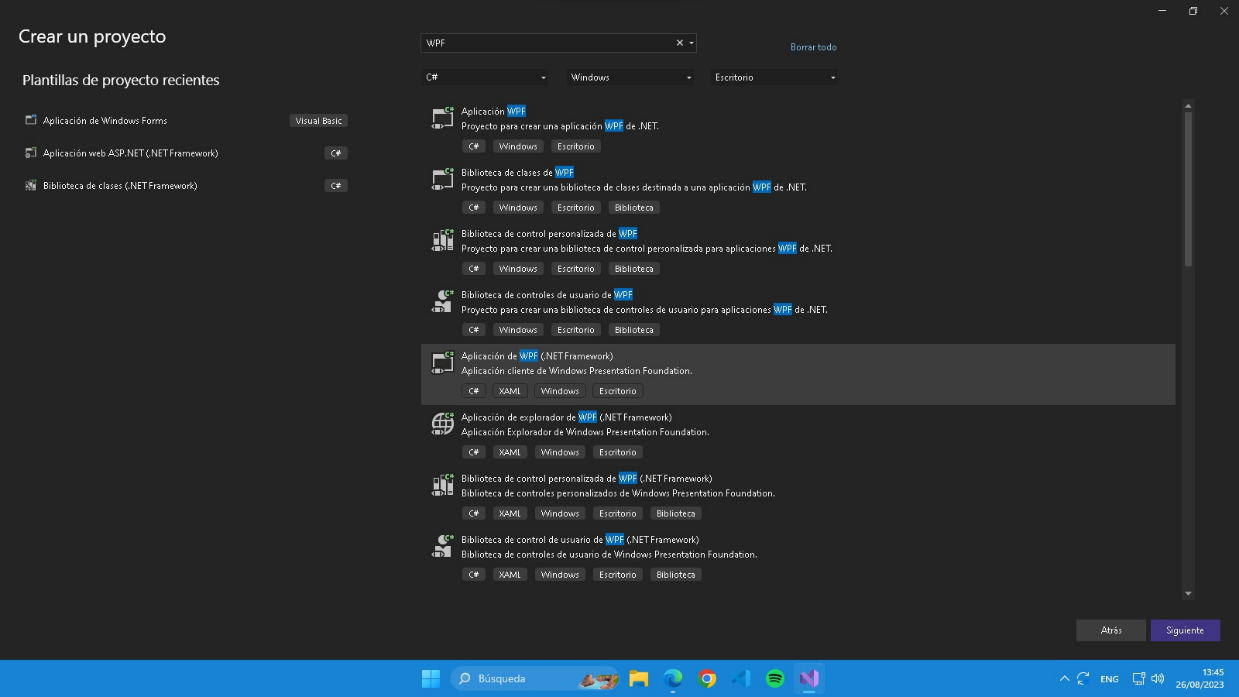


Figura 3. Selección del tipo de aplicación a desarrollar

Le da el siguiente nombre IntroduccionWPF\_Grupo# y en la opción que dice localización del proyecto presiona buscar y selecciona la carpeta “Aplicativo” que fue creada con anterioridad. En la versión de .NET Framework seleccionar la versión 4.7.2 para tener la máxima compatibilidad.

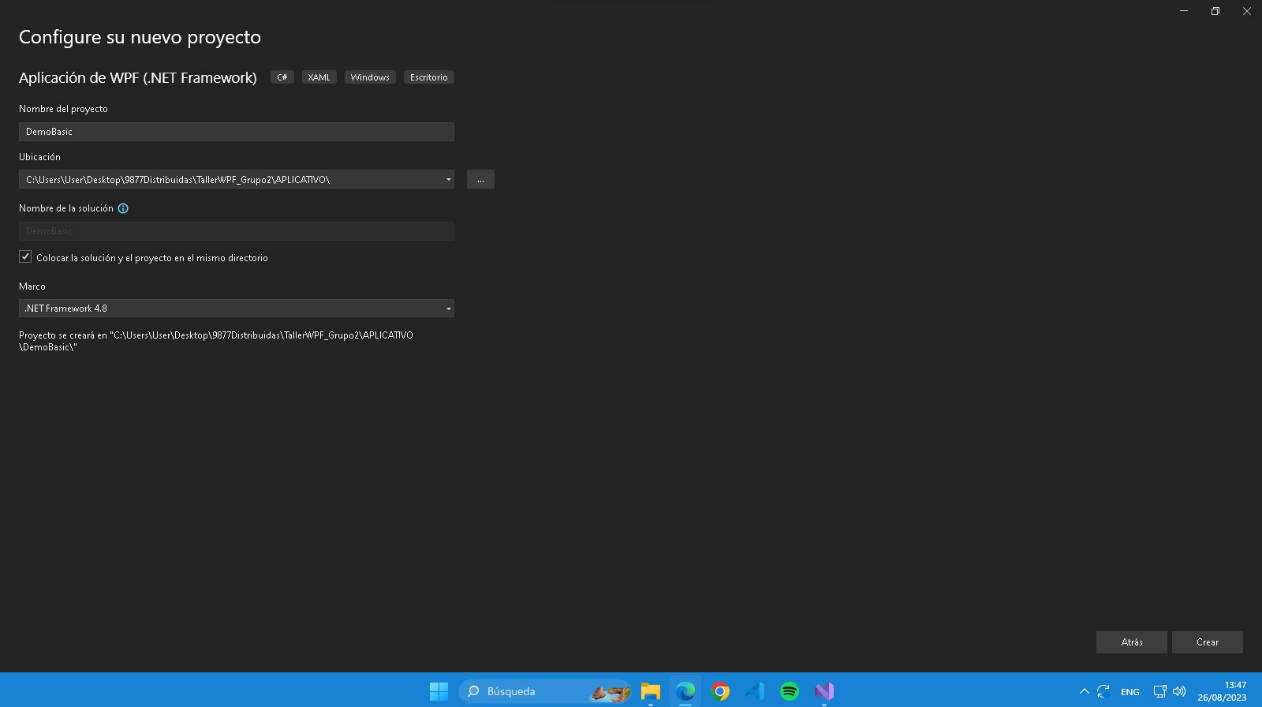


Figura 4. Selección del nombre y directorio donde guardar el proyecto.

De clic en crear y Verifique que la estructura del proyecto se encuentre de la siguiente manera.

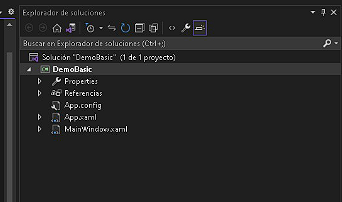


Figura 5. Estructura del proyecto generado.

## 4.1.2 CREACIÓN DE LA INTERFAZ PRINCIPAL DEL PROYECTO.

Debemos constatar que la aplicación tenga dos ventanas, la interfaz gráfica y la línea de código “Window” de la siguiente forma:

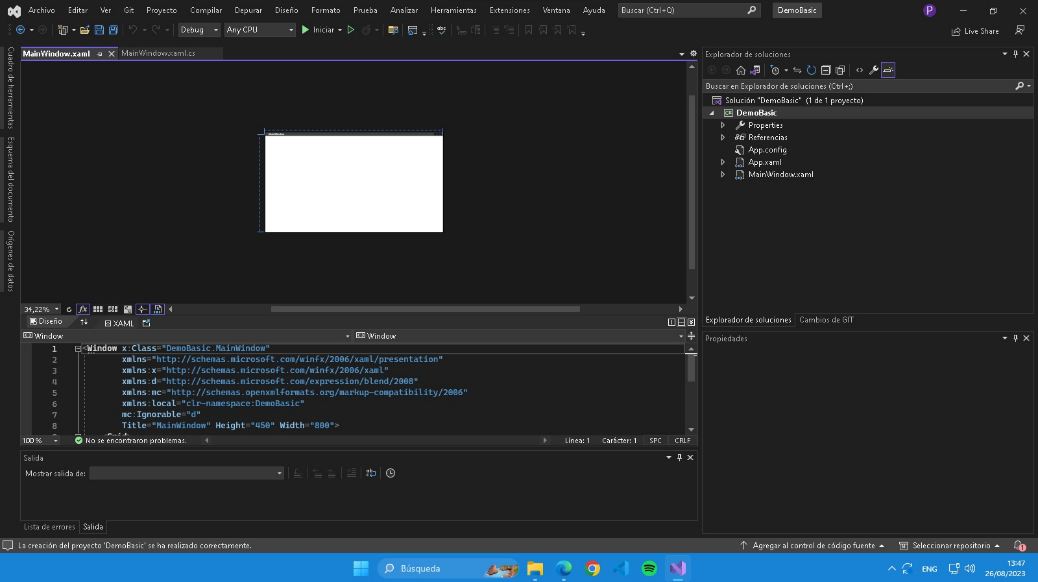


Figura 6. Estructura del proyecto.

Adicional, de click derecho sobre la ventana principal del proecto y agregue una nueva carpeta llamada “recurso” e importe su imagen favorita para el botón de usuario:

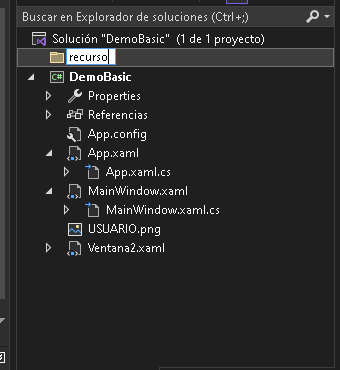


Figura 7. Crear carpeta imagen

Una vez verificada la estructura del proyecto e importado los recursos, procedemos a colocar el siguiente código:

Tabla . Window Principal

<Window x:Class="DemoBasic.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:DemoBasic"

mc:Ignorable="d"

Topmost="True"

Title="Ventana Principal" Height="400" Width="600">

<Grid>

<Button HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click">

<StackPanel Orientation="Horizontal">

<Image Margin="5"

Width="48"

Height="48"

Source="/USUARIO.png">

</Image>

<Label FontSize="20"

Content="Acceso al Sistema"

VerticalAlignment="Center"/>

</StackPanel>

</Button>

</Grid>

</Window>

Cuando el código este dentro del panel Window, proceda a resaltar todo el evento del botón y aplaste la tecla F12, esto lo llevara a la ventana MainWindow.xaml.cs, que controla la lógica principal del aplicativo después del evento y pegue el siguiente código:

Tabla . Codigo Window Principal Metodos

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace DemoBasic

{

/// <summary>

/// Lógica de interacción para MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

var textEnvio = "Hola Mundo desde WPF";

var nuevaVentana = new Ventana2(textEnvio);

nuevaVentana.Show();

}

}

}

# 4.2 CREACIÓN DE VENTANA RECEPTORA

### 4.2.1 CREACIÓN DE LA NUEVA VENTANA.

Diríjase al proyecto, seleccione la clase principal y con click izquierdo abra el panel de opciones, escoja la opción agregar y seleccione “nuevo elemento”:

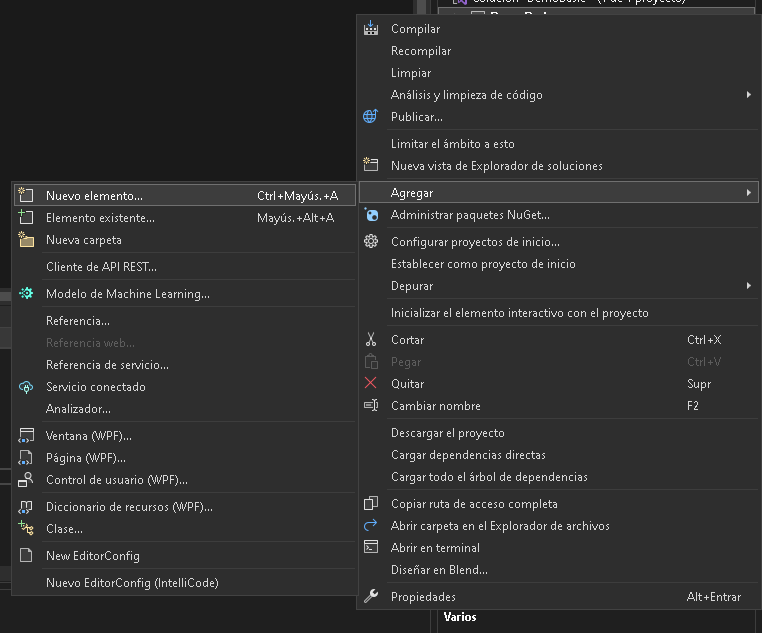


Figura 8. Agregar Nueva Ventana

Una vez agregada la Nueva Ventana con el nombre “Ventana2”, procedemos a agregar el siguiente código en el apartado Window de Ventana 2:

Tabla . Ventana2

<Window x:Class="DemoBasic.Ventana2"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:DemoBasic"

mc:Ignorable="d"

Title="Ventana2" Height="450" Width="800" Topmost="True">

<Grid>

<Label x:Name="lblTitle" HorizontalAlignment="Center" VerticalAlignment="Center"> </Label>

</Grid>

</Window>

# EJECUCIÓN DEL PROYECTO

De clic al botón Run para compilar el proyecto.

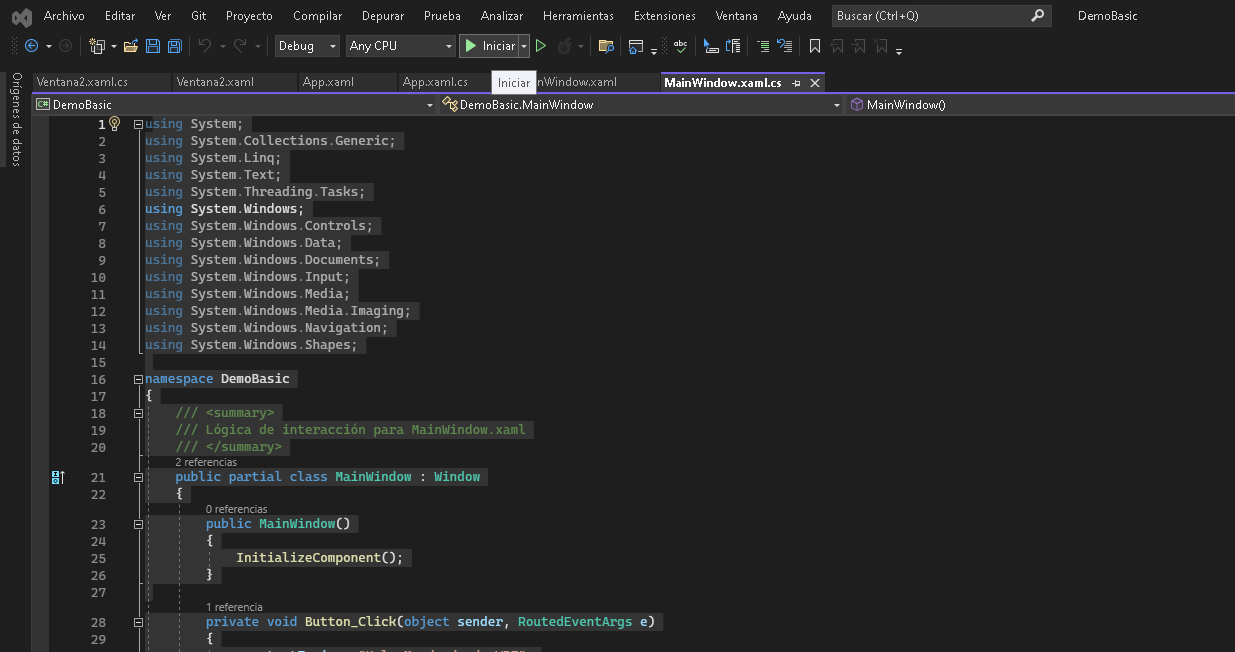


Figura 9. Botón para ejecutar el proyecto.

Al compilar el proyecto dará la siguiente vista.

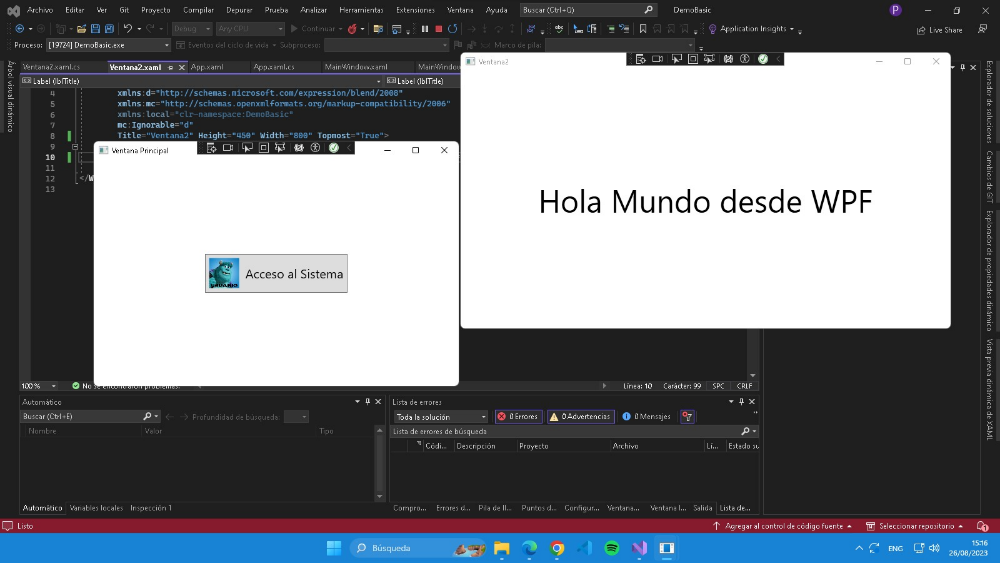


Figura 10. Ejecución del proyecto.

El método Button\_Click se ejecuta cuando se hace clic en el botón de la ventana principal. Dentro de este método, se crea una cadena textEnvio con el valor "Hola Mundo desde WPF". Luego, se crea una nueva instancia de Ventana2 pasando la cadena textEnvio como argumento al constructor. Finalmente, se muestra la nueva ventana (nuevaVentana.Show()).

Definición XAML de la Ventana Principal:

La ventana principal se define en XAML. Contiene un botón en un Grid que alberga un StackPanel. El StackPanel contiene una imagen y una etiqueta. Al hacer clic en el botón, se invoca el método Button\_Click definido en el código C# de la clase MainWindow.

Definición XAML de la Ventana2:

La ventana Ventana2 también se define en XAML. Contiene un Label llamado lblTitle centrado tanto horizontal como verticalmente. Esta ventana se utiliza para mostrar el título pasado como argumento en el constructor al crear una nueva instancia de Ventana2 en el método Button\_Click.

Funcionamiento de la Aplicación:

Cuando se inicia la aplicación, se muestra la ventana principal con un botón que tiene una imagen y una etiqueta. Al hacer clic en el botón, se crea una nueva instancia de Ventana2 y se muestra en la pantalla. La etiqueta en la ventana Ventana2 muestra el título "Hola Mundo desde WPF" que se pasó como argumento al crear la ventana. En resumen, esta aplicación demuestra cómo se puede manejar la interacción entre ventanas y pasar información de una ventana a otra en una aplicación WPF.

# CONCLUSIONES

* Interfaz Atractiva y Dinámica: El uso de recursos gráficos como imágenes e iconos en la interfaz de usuario de la ventana principal enriquece la apariencia visual de la aplicación. La combinación de una imagen y una etiqueta dentro de un botón no solo aporta estética, sino que también comunica de manera efectiva la acción que se puede realizar.
* Interacción entre Ventanas: El código muestra cómo crear una interacción fluida entre ventanas en una aplicación WPF. Al hacer clic en el botón de la ventana principal, se instancia y muestra una segunda ventana (Ventana2) con un título específico. Esto demuestra la capacidad de WPF para manejar múltiples ventanas y transmitir información entre ellas.
* Separación de Lógica y Diseño: El uso de XAML para definir la interfaz de usuario y C# para manejar la lógica de la aplicación ejemplifica la separación efectiva entre la presentación visual y la funcionalidad subyacente. Esto permite que diseñadores y desarrolladores trabajen de manera colaborativa, ya que pueden centrarse en áreas específicas sin interferir en el trabajo del otro.
* Mejora de la Usabilidad: La creación de ventanas adicionales, como Ventana2, puede enriquecer la experiencia del usuario al mostrar información contextual o proporcionar funcionalidades específicas. En este caso, la capacidad de mostrar un mensaje personalizado en una ventana separada agrega valor a la interacción del usuario y mejora la usabilidad global de la aplicación.

# RECOMENDACIONES

* Mantenimiento de Código Limpiamente Separado:

Es importante mantener una separación clara entre el código C# que maneja la lógica de la aplicación y el código XAML que define la interfaz de usuario. Para garantizar la legibilidad y el mantenimiento a largo plazo, considera utilizar patrones de diseño como el patrón Modelo-Vista-Controlador (MVC) o el patrón Modelo-Vista-Vista-Modelo (MVVM). Estos patrones ayudarán a organizar tu código de manera más eficiente, facilitando las actualizaciones y mejoras futuras.

* Optimización de Recursos Gráficos:

Si planeas usar imágenes y otros recursos gráficos en tu aplicación, asegúrate de optimizarlos para un rendimiento óptimo. Las imágenes deben tener tamaños adecuados y resoluciones apropiadas para evitar cargas lentas y un uso excesivo de memoria. Además, considera utilizar formatos de imagen adecuados, como JPEG para fotografías y PNG para gráficos con transparencia. La optimización de recursos gráficos garantizará una experiencia de usuario fluida y una respuesta rápida de la aplicación.

# BIBLIOGRAFÍA

|  |  |
| --- | --- |
| [1] | J. D. Luján, «¿Cómo funcionan los hilos en programación?,» EDTEAM, [En línea]. Available: https://ed.team/blog/como-funcionan-los-hilos-en-programacion. [Último acceso: 13 01 2023]. |
| [2] | «Estados de un hilo,» Iniciativa open source, [En línea]. Available: https://javaparajavatos.wordpress.com/2017/05/07/estados-de-un-hilo/. [Último acceso: 13 01 2023]. |
| [3] | «¿Qué es la concurrencia?,» theastrology, 2022. [En línea]. Available: https://es.theastrologypage.com/concurrency. [Último acceso: 13 01 2023]. |
| [4] | «Programación Paralela,» github ayudas, [En línea]. Available: http://ferestrepoca.github.io/paradigmas-de-programacion/paralela/paralela\_teoria/index.html. [Último acceso: 13 01 2023]. |
| [5] | G. Distel, «Hilos y concurrencia,» Universidad Nacional del Sur, [En línea]. Available: https://cs.uns.edu.ar/~gd/soyd/clasesgus/04-HilosyConcurrencia4x.pdf. [Último acceso: 13 01 2023]. |
| [6] | «Monitor Clase,» Microsoft, [En línea]. Available: https://learn.microsoft.com/es-es/dotnet/api/system.threading.monitor?view=net-7.0. [Último acceso: 13 01 2023]. |
| [7] | P. Echarte, «CREACIÓN DE HILOS CON PARAMETROS EN C#,» 16 01 2006. [En línea]. Available: http://www.eslomas.com/2006/01/creacion-hilos-con-parametros-csharp/. [Último acceso: 16 01 2023]. |
| [8] | A. Walton, «Sincronización de Hilos en Java,» JAVA DESDE CERO, [En línea]. Available: https://javadesdecero.es/avanzado/sincronizacion-de-hilos/#:~:text=Al%20usar%20m%C3%BAltiples%20hilos%2C%20a,se%20llama%20sincronizaci%C3%B3n%20(synchronization).. [Último acceso: 16 01 2023]. |
| [9] | B. Wagner, «instrucción lock: sincronización del acceso de subprocesos a un recurso compartido,» Microsoft Learn, 02 12 2022. [En línea]. Available: https://learn.microsoft.com/es-es/dotnet/csharp/language-reference/statements/lock. [Último acceso: 15 01 2023]. |
| [10] | «Comunicación entre threads,» Ediciones Eni, [En línea]. Available: https://www.ediciones-eni.com/open/mediabook.aspx?idR=929ccc4af4ac0368e9749e1672377f8e. [Último acceso: 15 01 2023]. |
| [11] | A. Ivonne, «Uso de threads en C#,» Slideshare, 05 04 2015. [En línea]. Available: https://es.slideshare.net/anaivonne33/uso-de-threads. [Último acceso: 15 01 2023]. |
| [12] | Microsoft, «Thread Clase (System.Threading),» Microsoft, [En línea]. Available: https://learn.microsoft.com/es-es/dotnet/api/system.threading.thread?view=net-7.0. [Último acceso: 15 1 2023]. |